BiCGCR2: A new extension of conjugate residual method for solving non-Hermitian linear systems

Xian-Ming Gu[1,2*], Ting-Zhu Huang[1†], Bruno Carpentieri[3‡]

1. School of Mathematical Sciences,
University of Electronic Science and Technology of China, Chengdu, Sichuan 611731, P.R. China
2. Institute of Mathematics and Computing Science,
University of Groningen, Nijenborgh 9, P.O. Box 407, 9700 AK Groningen, The Netherlands
3. School of Science and Technology,
Nottingham Trent University, Clifton Campus, Nottingham, NG11 8NS, UK

Abstract

In the present paper, we introduce a new extension of the conjugate residual (CR) for solving non-Hermitian linear systems with the aim of developing an alternative basic solver to the established biconjugate gradient (BiCG), biconjugate residual (BiCR) and biconjugate A-orthogonal residual (BiCOR) methods. The proposed Krylov subspace method, referred to as the BiCGCR2 method, is based on short-term vector recurrences and is mathematically equivalent to both BiCR and BiCOR. We demonstrate by extensive numerical experiments that the proposed iterative solver has often better convergence performance than BiCG, BiCR and BiCOR. Hence, it may be exploited for the development of new variants of non-optimal Krylov subspace methods.

Key words: BiCG; BiCR; Krylov subspace methods; Non-Hermitian linear systems; Bi-Lanczos procedure; Coupled two-term recurrences.

AMS Classification: 65F12; 65L05; 65N22.

1 Introduction

The core of many scientific computing and engineering simulations requires to solve large and sparse linear systems of the form

$$A\boldsymbol{x} = \boldsymbol{b}, \qquad A \in \mathbb{C}^{N\times N}, \quad \boldsymbol{b} \in \mathbb{C}^{N}, \tag{1}$$

where A is a non-Hermitian and possibly indefinite matrix, and $\boldsymbol{b}$ is the right-hand side vector. Numerical methods for solving system (1) on modern computers fall mainly into two categories: (sparse) direct and iterative solvers. Sparse direct solvers [1] are generally very accurate, robust and predictable in terms of both storage and algorithmic cost. Nevertheless, they tend to be too expensive to use for solving large-scale problems especially in terms of memory. Iterative solvers, namely the well-known class of Krylov subspace methods, can be an attractive alternative to

[*] *E-mail address:* guxianming@live.cn, x.m.gu@rug.nl
[†] Corresponding author. *E-mail address:* tingzhuhuang@126.com. Tel.: 86-28-61831608.
[‡] *E-mail address:* bcarpentieri@gmail.com

direct methods as they only require matrix-vector multiplications; see e.g. [2–5] and references therein. However, they generally lack robustness. It remains a research question to determine the classes of problems for which one algorithm is more efficient than others.

The conjugate gradient (CG) method [6] may be considered the Krylov method of choice in the case of Hermitian positive definite A. If A is complex symmetric (but non-Hermitian), i.e. $A \neq A^H$ but $A = A^T$, this property can be exploited in the design of the Krylov method; see e.g. our recent work [7] about the SCBiCG class of iterative algorithms. For indefinite A, the minimum residual (MINRES) method [8] and the conjugate residual (CR) method [9, 10] both enjoy attractive minimum norm residual property at each iteration step. Generalizations of the CG, MINRES, and CR methods have been proposed for solving non-Hermitian linear systems, such as BiCG [11, 12], FOM [2, pp. 165-168], GMRES [2, 13, pp. 172-180], GCR [14], BiCR [15, 16] and BiCOR [17, 18].

The choice of the Krylov algorithm is much less clear for non-Hermitian linear systems than for the Hermitian positive definite case. The GMRES and GCR methods enjoy an attractive minimum norm residual property that produce their typical smooth convergence behavior. However, they are based on the Arnoldi procedure [2, pp. 160-165], meaning that their computational and memory costs increase linearly with the number of iterations. The problem of cost may be overcome by restarting the iterative procedure after each cycle of, say, m iterations. However, the restarted GMRES and GCR methods, denoted as GMRES(m) [2, 13, pp. 179-180] and GCR(m) [14], respectively, lose any optimality property and they often suffer from slow convergence on difficult problems. On the other hand, since BiCG is based on the Bi-Lanczos procedure [2, 11, 12, pp. 229-233], it has constant computational work and low memory requirements per iteration step. Analogously, the BiCR and BiCOR methods can be derived from the so-called biconjugate A-orthonormalizaion procedure [16, 17], which is similar to the classical Bi-Lanczos process. Hence they are based on short-term vector recurrences, and require only $\mathcal{O}(N)$ extra storage in addition to the matrix and $\mathcal{O}(N)$ operations for solving the system.

Although non-optimal Krylov methods based on short-term vector recurrences tend to exhibit irregular convergence behavior, their limited cost has motivated and still motivates a growing interest in improving their performance. In 1989, Sonneveld [19] established the first successful variant of BiCG, referred to as the CGS method. Later, van der Vorst [20] derived one of the most successful variants of BiCG, known as the BiCGSTAB method. Based on the ideas behind the development of BiCGSTAB and CGS, a lot of generalized iterative solvers have been proposed such as BiCGSTAB2 [21], BiCGSTAB(ℓ) [22, 23], GCGS [24] and GPBiCG [25]. The quasi-minimal residual (QMR) [26] method, that is closely related to BiCG, is an attractive variant because it displays (quasi)-smoother convergence behavior than BiCG, it can remedy pivot breakdown and may avoid Bi-Lanczos breakdown by a look-ahead strategy (e.g. refer to [27]). A transpose-free variant of QMR, called the TFQMR method [28], and a hybrid of TFQMR and BiCGSTAB, called the QMRCGSTAB method [29], have been also proposed. For further discussion of this topic, one can refer to an excellent survey paper in this area [4]. Additionally, the efficient short-recurrence IDR(s) method is proposed recently by Sonneveld and van Gijzen in [30], which is closely related to the other interesting BiCG-type variant-ML(k)BiCGSTAB method by Yeung and Chan [31]. Several reported experiments show that these two methods can be efficient tools for solving non-Hermitian linear systems [30, 31].

Along the same lines of development of hybrid BiCG methods, various hybrid BiCR methods, such as CRS, BiCRSTAB, BiCRSTAB(ℓ) and GPBiCR have been proposed for solving non-Hermitian linear systems, refer to [16,32] for details. At almost the same time, our research group

also developed some efficient hybrid BiCOR variants, including CORS [17, 18], GCORS [33], BiCORSTAB [17], BiCORSTAB2 [34] and GPBiCOR [35]. Many numerical experiments on practical applications have illustrated the robustness of the hybrid BiCR and hybrid BiCOR methods; refer, e.g., to [16–18, 32, 35, 36] for details.

The earlier discussion highlights the important role that the BiCG, BiCR and BiCOR method play in the developments of hybrid Lanczos-type variants. Furthermore, BiCG is closely related to the QMR method [37] and, similarly, BiCR and BiCOR are closely related to QMOR [38]. In this study, we propose a novel basic iterative scheme derived from short-term vector recurrences, that can be seen as an extension of the CR method to non-Hermitian linear systems, for the development of non-optimal Krylov subspace methods.

The rest of this paper is organized as follows. In Section 2, we first review the development of extensions of the CR method for solving complex symmetric and non-Hermitian linear systems. Then we recall the underlying relations [7,15,16,39] among the COCR [39], BiCGCR [40,41] and BiCR [15] methods. Based on the above analysis, a new extension of the CR method, named the BiCGCR2 method, and its preconditioned version, are derived. We also discuss some properties of the proposed BiCGCR2 method. In Section 3 we prove that the preconditioned BiCGCR2 (PBiCGCR2) is mathematically equivalent to the preconditioned BiCR (PBiCR), and then we derive a relation between BiCGCR2 and some related Krylov subspace methods. In Section 4, extensive numerical experiments are reported to illustrate the effectiveness of the proposed method. Finally, the paper closes with some conclusions in Section 5.

Throughout this paper, A^H denotes the conjugate transpose of A, $(\boldsymbol{x}, \boldsymbol{y})$ denotes the dot product given by $\boldsymbol{x}^H\boldsymbol{y}$, and we use the notation

$$\mathcal{K}_n(A, \boldsymbol{r}_0) := \text{span}\{\boldsymbol{r}_0, A\boldsymbol{r}_0, \ldots, A^{n-1}\boldsymbol{r}_0\}$$

for the n-dimensional Krylov subspace generated by A and initial residual vector $\boldsymbol{r}_0$.

2 The derivation of the BiCGCR2 method

Sogabe, Sugihara and Zhang have extended the CR method to the COCR method and the BiCR method; refer to [15, 16, 39] for solving complex symmetric and non-Hermitian linear systems, respectively. The COCR method is a special case of the BiCR method. Additionally, since the BiCR method was proposed, this method was also improved (or modified) for various systems of linear equations involving non-Hermitian coefficient matrices, e.g., refer to [42–44] for details. In previous work, we have proved that it is mathematically equivalent to the BiCGCR algorithm proposed by Clemens in [41], the difference lying only in the choice of the scalar factors α_k and β_k within the inner iteration loop; refer to [7, 16] for details. The relations between the BiCR method and the COCR method naturally leads to extend the BiCGCR method to a new variant named BiCGCR2 for solving non-Hermitian linear systems. The pseudo code of the BiCGCR2 method is sketched in Algorithm 1.

Note that in Algorithm 1, $A\boldsymbol{p}_n = A\boldsymbol{r}_n + \beta_{n-1}A\boldsymbol{p}_{n-1}$ is newly added to reduce the number of matrix-vector multiplications at each iteration step. The theoretical results of the complex BiCG method transfer directly to BiCGCR2. The iterative procedure of BiCGCR2 is governed by a Petrov-Galerkin condition

$$\boldsymbol{r}_n = \boldsymbol{b} - A\boldsymbol{x}_n \perp \mathcal{L}_n \quad \text{with} \quad \boldsymbol{x}_n \in \boldsymbol{x}_0 + \mathcal{K}_n. \tag{2}$$

Algorithm 1 Algorithm of the BiCGCR2 method

$\boldsymbol{x}_0$ is an initial guess, $\boldsymbol{r}_0 = \boldsymbol{b} - A\boldsymbol{x}_0$.
Choose $\boldsymbol{r}_0^*$ (for example, $\boldsymbol{r}_0^* = \boldsymbol{r}_0$),
Set $\boldsymbol{p}_{-1}^* = \boldsymbol{p}_{-1} = \boldsymbol{0}$, $\beta_{-1} = 0$,
for $n = 0, 1, \ldots,$ until convergence **do**
  $\boldsymbol{p}_n = \boldsymbol{r}_n + \beta_{n-1}\boldsymbol{p}_{n-1}$,
  $\boldsymbol{p}_n^* = \boldsymbol{r}_n^* + \bar{\beta}_{n-1}\boldsymbol{p}_{n-1}^*$,
  $(A\boldsymbol{p}_n = A\boldsymbol{r}_n + \beta_{n-1}A\boldsymbol{p}_{n-1},)$
  $\alpha_n = \frac{\langle A^H\boldsymbol{p}_n^*, \boldsymbol{r}_n\rangle}{\langle A^H\boldsymbol{p}_n^*, A\boldsymbol{p}_n\rangle}$,
  $\boldsymbol{x}_{n+1} = \boldsymbol{x}_n + \alpha_n\boldsymbol{p}_n$,
  $\boldsymbol{r}_{n+1} = \boldsymbol{r}_n - \alpha_n A\boldsymbol{p}_n$,
  $\boldsymbol{r}_{n+1}^* = \boldsymbol{r}_n^* - \bar{\alpha}_n A^H\boldsymbol{p}_n^*$,
  $\beta_n = -\frac{\langle A^H\boldsymbol{p}_n^*, A\boldsymbol{r}_{n+1}\rangle}{\langle A^H\boldsymbol{p}_n^*, A\boldsymbol{p}_n\rangle}$.
end for

with respect to the search subspace $\mathcal{K}_n$ and constraints subspace $\mathcal{L}_n = \mathcal{L}_n(A^H, \boldsymbol{r}_0^*)$, where one has $\boldsymbol{r}_0^* = A^H\boldsymbol{r}_0^*$. In [7,40,41,45], the BiCGCR method is shown to coincide with the CR method of Stiefel [9] for real symmetric problems, which has a residual minimization property $\|\boldsymbol{r}_k\|_2$. This may explain the smaller oscillations that are typically observed in the residual norm for the BiCGCR2 method compared to the BiCG method. We see from Algorithm 1 that the approximate solution $\boldsymbol{x}_n$ can be generated by coupled two-term recurrences. If the coefficient matrix is Hermitian, then BiCGCR2 reduces to CR.

Next, we can obtain some properties of BiCGCR2 that suggest another derivation. For simplicity, in the following discussion we assume that the coefficient matrix is real nonsymmetric, i.e., $A \neq A^T$. Observing Algorithm 1, we see that the four iterates $\boldsymbol{r}_n, \boldsymbol{p}_n, \boldsymbol{r}_n^*$, and $\boldsymbol{p}_n^*$ can be expressed as

$$\boldsymbol{r}_n = R_n(A)\boldsymbol{r}_0, \quad \boldsymbol{p}_n = P_n(A)\boldsymbol{r}_0, \tag{3}$$
$$\boldsymbol{r}_n^* = R_n(A^T)\boldsymbol{r}_0, \quad \boldsymbol{p}_n^* = P_n(A^T)\boldsymbol{r}_0^*, \tag{4}$$

where R_n and P_n are polynomials of degree n satisfying

$$\begin{aligned} &R_0(\lambda) = 1, \qquad P_0(\lambda) := 1, \\ &R_n(\lambda) = R_{n-1}(\lambda) - \alpha_{n-1}\lambda P_{n-1}(\lambda), \\ &P_n(\lambda) = R_n(\lambda) + \beta_{n-1}P_{n-1}(\lambda), \quad \text{for } n = 1, 2, \ldots. \end{aligned}$$

As seen from (3)-(4) and from Algorithm 1, the following results are obtained if no breakdown occurs:

Theorem 1 *For $i \neq j$, the following bi-orthogonality properties hold:*

$$(\boldsymbol{r}_i^*, A\boldsymbol{r}_j) = 0, \tag{5}$$
$$(A^H\boldsymbol{p}_i^*, A\boldsymbol{p}_j) = 0. \tag{6}$$

Proof. It follows from (3) and (4) that $\langle \boldsymbol{r}_i^*, A\boldsymbol{r}_j\rangle = \langle R_i(A^T)\boldsymbol{r}_0^*, AR_j(A)\boldsymbol{r}_0\rangle = \langle R_j(A^T)\boldsymbol{r}_0^*, A$ $R_i(A)\boldsymbol{r}_0\rangle = \langle \boldsymbol{r}_j^*, A\boldsymbol{r}_i\rangle$. Similarly, from (6) we obtain $\langle A^T\boldsymbol{p}_i^*, A\boldsymbol{p}_j\rangle = (A^T\boldsymbol{p}_j^*, A\boldsymbol{p}_i)$. Hence, the statements of (5) and (6) are equivalent with

$$\langle \boldsymbol{r}_i^*, A\boldsymbol{r}_j\rangle = 0 \ \text{ and } \ \langle A^T\boldsymbol{p}_i^*, A\boldsymbol{p}_j\rangle = 0, \quad \text{for all } j < i. \tag{7}$$

Now, we give a proof of (7) by induction. Since the trivial case $i = 1$ is obvious from Algorithm 1, we assume that property (7) holds for $j < i \leq k$. Then, we show that

$$\langle \boldsymbol{r}_{k+1}^*, A\boldsymbol{r}_j \rangle = 0, \tag{8}$$
$$\langle A^T\boldsymbol{p}_{k+1}^*, A\boldsymbol{p}_j \rangle = 0. \tag{9}$$

First, let us show (8). For the case $j < k$ it follows from the above assumption that

$$\begin{aligned}
\langle \boldsymbol{r}_{k+1}^*, A\boldsymbol{r}_j \rangle &= \langle \boldsymbol{r}_k^*, A\boldsymbol{r}_j \rangle - \alpha_k \langle A^T\boldsymbol{p}_k^*, A\boldsymbol{r}_j \rangle \\
&= -\alpha_k \langle A^T\boldsymbol{p}_k^*, A\boldsymbol{r}_j \rangle \\
&= -\alpha_k \langle A^T\boldsymbol{p}_k, A\boldsymbol{p}j \rangle - \alpha_k\beta_{j-1} \langle A^T\boldsymbol{p}_k^*, A\boldsymbol{p}_{j-1} \rangle \\
&= 0.
\end{aligned}$$

For the case $j = k$ we obtain

$$\begin{aligned}
\langle \boldsymbol{r}_{k+1}^*, A\boldsymbol{r}_k \rangle &= \langle \boldsymbol{r}_k^*, A\boldsymbol{r}_k) - \alpha_k \langle A^T\boldsymbol{p}_k^*, A\boldsymbol{r}_k \rangle \\
&= \langle \boldsymbol{r}_k^*, A\boldsymbol{r}_k \rangle - \alpha_k \langle A^T\boldsymbol{p}_k^*, A\boldsymbol{p}_k \rangle - \alpha_k\beta_{k-1} \langle A^T\boldsymbol{p}_k^*, A\boldsymbol{p}_{k-1} \rangle \\
&= \langle \boldsymbol{r}_k^*, A\boldsymbol{r}_k \rangle - \alpha_k \langle A^T\boldsymbol{p}_k^*, A\boldsymbol{p}_k \rangle \\
&= \langle \boldsymbol{p}_k^* - \beta_{k-1}\boldsymbol{p}_{k-1}^*, A\boldsymbol{r}_k \rangle - \alpha_k \langle A^T\boldsymbol{p}_k^*, A\boldsymbol{p}_k \rangle \\
&= -\beta_{k-1} \langle \boldsymbol{p}_{k-1}^*, A(\boldsymbol{r}_{k-1} - \alpha_{k-1}A\boldsymbol{p}_{k-1} \rangle \\
&= -\beta_{k-1} \langle A^T\boldsymbol{p}_{k-1}^*, \boldsymbol{r}_{k-1} \rangle + \beta_{k-1}\alpha_{k-1} \langle A^T\boldsymbol{p}_{k-1}^*, A\boldsymbol{p}_{k-1} \rangle \\
&= 0
\end{aligned}$$

from the computational formulas of α_k in line 8 (of Algorithm 1) and β_k in line 12 (of Algorithm 1). Next, we show (9). For the case $j < k$, it follows from the first result of the proof that

$$\langle A^T\boldsymbol{p}_{k+1}^*, A\boldsymbol{p}_j \rangle = \langle A^T\boldsymbol{r}_{k+1}^*, A\boldsymbol{p}_j \rangle + \beta_k \langle A^T\boldsymbol{p}_k^*, A\boldsymbol{p}_j \rangle = \frac{1}{\alpha_j}(A^T\boldsymbol{r}_{k+1}^*, \boldsymbol{r}_j - \boldsymbol{r}_{j+1}) = 0.$$

For the case $j = k$, we obtain

$$\begin{aligned}
\langle A^T\boldsymbol{p}_{k+1}^*, A\boldsymbol{p}_k \rangle &= \langle A^T\boldsymbol{r}_{k+1}^*, A\boldsymbol{p}_k \rangle + \beta_k \langle A^T\boldsymbol{p}_k^*, A\boldsymbol{p}_k \rangle \\
&= \frac{1}{\alpha_k} \langle A^T\boldsymbol{r}_{k+1}^*, \boldsymbol{r}_k - \boldsymbol{r}_{k+1} \rangle + \beta_k \langle A^T\boldsymbol{p}_k^*, A\boldsymbol{p}_k \rangle \\
&= -\frac{1}{\alpha_k} \langle A^T\boldsymbol{r}_{k+1}^*, \boldsymbol{r}_{k+1} \rangle + \beta_k \langle A^T\boldsymbol{p}_k^*, A\boldsymbol{p}_k \rangle \\
&= -\frac{1}{\alpha_k} \langle \boldsymbol{r}_k^* - \alpha_k A^T\boldsymbol{p}_k^*, A\boldsymbol{r}_{k+1} \rangle + \beta_k \langle A^T\boldsymbol{p}_k^*, A\boldsymbol{p}_k \rangle \\
&= -\frac{1}{\alpha_k} \langle \boldsymbol{r}_k^*, A\boldsymbol{r}_{k+1} \rangle + \langle A^T\boldsymbol{p}_k^*, A\boldsymbol{r}_{k+1} \rangle + \beta_k \langle A^T\boldsymbol{p}_k^*, A\boldsymbol{p}_k \rangle \\
&= 0
\end{aligned}$$

from the formulas of α_k and β_k at lines 8 and 12 of Algorithm 1, respectively. □

Corollary 1 *Some further properties of BiCGCR2 are*

$$\langle \boldsymbol{r}_i^*, A\boldsymbol{p}_j\rangle = 0 \quad \text{for } i > j, \tag{10}$$

$$\langle \boldsymbol{r}_i^*, A\boldsymbol{r}_i\rangle = \langle \boldsymbol{r}_i^*, A\boldsymbol{p}_i\rangle, \tag{11}$$

$$\langle A^T\boldsymbol{r}_i^*, A\boldsymbol{p}_i\rangle = \langle A^T\boldsymbol{p}_i^*, A\boldsymbol{p}_i\rangle. \tag{12}$$

Proof. First, we give a proof of (10). From the recurrence in line 5 (of Algorithm 1) it follows that $\langle \boldsymbol{r}_i^*, A\boldsymbol{p}_j\rangle = \langle \boldsymbol{r}_i^*, A\boldsymbol{r}_j\rangle + \beta_{j-1}\langle \boldsymbol{r}_i^*, A\boldsymbol{p}_{j-1}\rangle$, and thus from property (5) we obtain $\langle \boldsymbol{r}_i^*, A\boldsymbol{p}_j\rangle = \beta_{j-1}\langle \boldsymbol{r}_i^*, A\boldsymbol{p}_{j-1}\rangle$. Applying this process recursively, we finally obtain $\langle \boldsymbol{r}_i^*, A\boldsymbol{p}_j\rangle = \beta_{j-1}\beta_{j-2}\cdots\beta_0\langle \boldsymbol{r}_i^*, A\boldsymbol{p}_0\rangle$. Hence, from $\boldsymbol{p}_0 = \boldsymbol{r}_0$ and (5), property (10) is naturally followed.

Second, we give a proof of (11). From the recurrence in line 5 (of Algorithm 1) it follows that $\langle \boldsymbol{r}_i^*, A\boldsymbol{r}_i\rangle = \langle \boldsymbol{r}_i^*, A\boldsymbol{p}_i\rangle - \beta_{i-1}\langle \boldsymbol{r}_i^*, A\boldsymbol{p}_{i-1}\rangle$. Since the second term is zero by (10), the property (11) is immediately established.

Finally, we present a proof of (12). According to the recurrence in line 6 (of Algorithm 1) it follows that $\langle A^T\boldsymbol{r}_i^*, A\boldsymbol{p}_i\rangle = \langle A^T\boldsymbol{p}_i^*, A\boldsymbol{p}_i\rangle - \beta_{i-1}\langle A^T\boldsymbol{p}_{i-1}^*, A\boldsymbol{p}_i\rangle$. Since the second term is zero from (6), property (12) is established. □

Furthermore, if we employ the same lines of development of the preconditioned CR (PCR) method, the following preconditioned version of BiCGCR2 can be immediately derived. The pseudo code of the resulting algorithm is given as follows

Algorithm 2 The preconditioned BiCGCR2 method (K is the preconditioner)

1: $\boldsymbol{x}_0$ is an initial guess, $\boldsymbol{r}_0 = \boldsymbol{b} - A\boldsymbol{x}_0$.
2: Choose $\boldsymbol{r}_0^*$ (for example, $\boldsymbol{r}_0^* = \boldsymbol{r}_0$),
3: Set $\boldsymbol{p}_{-1}^* = \boldsymbol{p}_{-1} = \boldsymbol{0}$, $\beta_{-1} = 0$,
4: **for** $n = 0, 1, \ldots,$ until convergence **do**
5: $\quad \boldsymbol{p}_n = K^{-1}\boldsymbol{r}_n + \beta_{n-1}\boldsymbol{p}_{n-1}$,
6: $\quad \boldsymbol{p}_n^* = K^{-H}\boldsymbol{r}_n^* + \bar{\beta}_{n-1}\boldsymbol{p}_{n-1}^*$,
7: $\quad (A\boldsymbol{p}_n = AK^{-1}\boldsymbol{r}_n + \beta_{n-1}A\boldsymbol{p}_{n-1},)$
8: $\quad \alpha_n = \frac{\langle A^H\boldsymbol{p}_n^*, K^{-1}\boldsymbol{r}_n\rangle}{\langle K^{-H}A^H\boldsymbol{p}_n^*, A\boldsymbol{p}_n\rangle}$,
9: $\quad \boldsymbol{x}_{n+1} = \boldsymbol{x}_n + \alpha_n\boldsymbol{p}_n$,
10: $\quad \boldsymbol{r}_{n+1} = \boldsymbol{r}_n - \alpha_n A\boldsymbol{p}_n$,
11: $\quad (K^{-H}\boldsymbol{r}_{n+1}^* = K^{-H}\boldsymbol{r}_n^* - \bar{\alpha}_n K^{-H}A^H\boldsymbol{p}_n^*)$,
12: $\quad \beta_n = -\frac{\langle K^{-H}A^H\boldsymbol{p}_n^*, AK^{-1}\boldsymbol{r}_{n+1}\rangle}{\langle K^{-H}A^H\boldsymbol{p}_n^*, A\boldsymbol{p}_n\rangle}$.
13: **end for**

Note that when the coefficient matrix A is Hermitian, Algorithm 2 reduces to PCR with the choice $\boldsymbol{r}_0^* = \boldsymbol{r}_0$, since in this case $\boldsymbol{r}_n^* = \boldsymbol{r}_n, \boldsymbol{p}_n^* = \boldsymbol{p}_n, \bar{\alpha}_n = \alpha_n$ and $\bar{\beta}_n = \beta_n$, see [7, 40]. However, the above version of PBiCGCR is more competitive than the one described in [40, 41] because it requires only one solution of the generalized residual equations

$$K\boldsymbol{z} = \boldsymbol{r}, \tag{13}$$

involving the preconditioner K, in the initialization procedure. When the coefficient matrix A is symmetric not Hermitian, i.e. $A = A^T \neq A^H$, we can derive a novel version of the preconditioned BiCGCR (PBiCGCR) method from Algorithm 2 with the choice $\boldsymbol{r}_0^* = \bar{\boldsymbol{r}}_0$, which results in $\boldsymbol{r}_n^* = \bar{\boldsymbol{r}}_n$ and $\boldsymbol{p}_n^* = \bar{\boldsymbol{p}}_n$; the pseudo code of PBiCGCR is given in Algorithm 3.

Algorithm 3 The preconditioned BiCGCR method (K is the preconditioner)

$\boldsymbol{x}_0$ is an initial guess, $\boldsymbol{r}_0 = \boldsymbol{b} - A\boldsymbol{x}_0$, solve $\boldsymbol{z}_0 = K^{-1}\boldsymbol{r}_0$,
Set $\boldsymbol{p}_{-1} = \boldsymbol{0}$, $\beta_{-1} = 0$, $\boldsymbol{q}_0 = A\boldsymbol{p}_0$, $\boldsymbol{s}_0 = A\boldsymbol{z}_0$
for $n = 0, 1, \ldots,$ until convergence **do**
  $\boldsymbol{p}_n = \boldsymbol{z}_n + \beta_{n-1}\boldsymbol{p}_{n-1}$,
  $\boldsymbol{q}_n = \boldsymbol{s}_n + \beta_{n-1}\boldsymbol{q}_{n-1}$,
  Solve $\boldsymbol{t}_n = K^{-1}\boldsymbol{q}_n$,
  $\alpha_n = \frac{\langle \bar{\boldsymbol{q}}_n, \boldsymbol{z}_n \rangle}{\langle \boldsymbol{t}_n, \boldsymbol{q}_n \rangle}$,
  $\boldsymbol{x}_{n+1} = \boldsymbol{x}_n + \alpha_n \boldsymbol{p}_n$,
  $\boldsymbol{r}_{n+1} = \boldsymbol{r}_n - \alpha_n \boldsymbol{q}_n$,
  $\boldsymbol{z}_{n+1} = \boldsymbol{z}_n - \alpha_n \boldsymbol{t}_n$,
  Compute $\boldsymbol{s}_{n+1} = A\boldsymbol{z}_{n+1}$,
  $\beta_n = -\frac{\langle \bar{\boldsymbol{t}}_n, \boldsymbol{s}_{n+1} \rangle}{\langle \boldsymbol{t}_n, \boldsymbol{q}_n \rangle}$.
end for

In Table 1 we analyze the computational cost for the proposed BiCGCR2 algorithm compared to BiCG, BiCR, BiCOR and QMR for solving linear system (1) using a preconditioner K (if available). Here "6 or 7" means "6" for the unpreconditioned BiCR/BiCOR/BiCGCR2 and "7" for their preconditioned versions. The BiCGCR2 method requires almost the same algorithmic cost per step (expect one more inner product) as other solvers. However, as we will illustrate by numerical experiments in Section 4, it often converges faster than BiCG, BiCR and BiCOR requiring slightly less number of iterations and less CPU elapsed time.

Table 1: Summary of algorithmic cost per iteration step

Method	$\langle \boldsymbol{x}, \boldsymbol{y} \rangle$	$\boldsymbol{y} = A\boldsymbol{x}$	$\boldsymbol{y} = A^H\boldsymbol{x}$	$\boldsymbol{y} = K^{-1}\boldsymbol{x}$	$\boldsymbol{y} = K^{-H}\boldsymbol{x}$	$\alpha\boldsymbol{x} + \boldsymbol{y}$
BiCG	2	1	1	1	1	5
BiCR	2	1	1	1	1	6 or 7
BiCOR	2	1	1	1	1	6 or 7
BiCGCR2	3	1	1	1	1	6 or 7
QMR	3	1	1	1	1	7

3 Mathematical equivalence of BiCGCR2 and BiCR

It has been shown by Sogabe and Zhang that the BiCGCR method is mathematically equivalent to the COCR method [39]. The difference lies in the choice of the coefficients α_k and β_k. However, they did not give a detailed proof of this relationship. Then, Gu *et al.* capitalized on these ideas and gave a proof of the mathematical equivalence between the PBiCGCR and PCOCR methods (see [7, 41] for details). As mentioned earlier in this article, the PBiCGCR and PCOCR methods are special cases of the PBiCGCR2 and PBiCR methods, respectively. Motivated by these considerations, we can also investigate the underlying relations between the PBiCGCR2 method and the PBiCR method. From the analysis of the following scalar

coefficients

$$\alpha_k^{PBiCGCR2} = \frac{(\boldsymbol{p}_k^*)^T AK^{-1}\boldsymbol{r}_k}{(\boldsymbol{p}_k^*)^T AK^{-1}A\boldsymbol{p}_k}, \tag{14}$$

$$\alpha_k^{PBiCR} = \frac{(\boldsymbol{r}_k^*)^T K^{-1}AK^{-1}\boldsymbol{r}_k}{(\boldsymbol{p}_k^*)^T AK^{-1}A\boldsymbol{p}_k} \tag{15}$$

and

$$\beta_k^{PBiCGCR2} = -\frac{(\boldsymbol{p}_k^*)^T AK^{-1}AK^{-1}\boldsymbol{r}_{k+1}}{(\boldsymbol{p}_k^*)^T AK^{-1}A\boldsymbol{p}_k}, \tag{16}$$

$$\beta_k^{PBiCR} = \frac{(\boldsymbol{r}_{k+1}^*)^T K^{-1}AK^{-1}\boldsymbol{r}_{k+1}}{(\boldsymbol{r}_k^*)^T K^{-1}AK^{-1}\boldsymbol{r}_k}, \tag{17}$$

we can obtain the following conclusions.

Theorem 2 *For all* $n \in \mathcal{N}$

$$\alpha_n^{PBiCGCR2} = \alpha_n^{PBiCR}.$$

Proof. The identity of the denominators in $\alpha_n^{PBiCGCR2}$ and α_n^{PBiCR} requires to show the identity

$$(\boldsymbol{p}_n^*)^T AK^{-1}\boldsymbol{r}_n = (\boldsymbol{r}_n^*)^T K^{-1}AK^{-1}\boldsymbol{r}_n \tag{18}$$

for all $n = 0, 1, 2, \ldots$. By rewriting

$$\begin{aligned}(\boldsymbol{p}_n^*)^T AK^{-1}\boldsymbol{r}_n &= (K^{-T}\boldsymbol{r}_n^* + \beta_{n-1}\boldsymbol{p}_{n-1}^*)^T AK^{-1}\boldsymbol{r}_n \\ &= (\boldsymbol{r}_n^*)^T K^{-1}AK^{-1}\boldsymbol{r}_n + \beta_{n-1}(\boldsymbol{p}_{n-1}^*)^T AK^{-1}\boldsymbol{r}_n,\end{aligned} \tag{19}$$

the identity (19) holds for $\beta_{n-1} \neq 0$, iff

$$(\boldsymbol{p}_{n-1}^*)^T AK^{-1}\boldsymbol{r}_n \equiv 0 \tag{20}$$

for all $n = 1, 2, \ldots$. The bi-orthogonality conditions of the preconditioned BiCR residuals hold in the case $n = 1$ from

$$(\boldsymbol{p}_0^*)^T AK^{-1}\boldsymbol{r}_1 = (K^{-T}\boldsymbol{r}_0^*)^T AK^{-1}\boldsymbol{r}_1 = (\boldsymbol{r}_0^*)^T K^{-1}AK^{-1}\boldsymbol{r}_1 = 0. \tag{21}$$

The identity (20) for the case $n + 1$ is results from

$$\begin{aligned}(\boldsymbol{p}_n^*)^T AK^{-1}\boldsymbol{r}_{n+1} &= (K^{-T}\boldsymbol{r}_n^* + \beta_{n-1}\boldsymbol{p}_{n-1}^*)^T AK^{-1}\boldsymbol{r}_{n+1} \\ &= (\boldsymbol{r}_n^*)^T K^{-1}AK^{-1}\boldsymbol{r}_{n+1} + \beta_{n-1}(\boldsymbol{p}_{n-1}^*)^T AK^{-1}\boldsymbol{r}_{n+1} \\ &= \beta_{n-1}(\boldsymbol{p}_{n-1}^*)^T AK^{-1}\boldsymbol{r}_{n+1} \\ &= \beta_{n-1}(\boldsymbol{p}_{n-1}^*)^T AK^{-1}(\boldsymbol{r}_n - \alpha_n A\boldsymbol{p}_n) \\ &= \beta_{n-1}(\boldsymbol{p}_{n-1}^*)^T AK^{-1}\boldsymbol{r}_n - \beta_{n-1}\alpha_n(K^{-T}A^T\boldsymbol{p}_{n-1}^*)^T A\boldsymbol{p}_n \\ &= 0,\end{aligned}$$

by induction from the case n and the bi-orthogonality relation of the PBiCR method $\langle K^{-T}A^T\boldsymbol{p}_{n-1}^*, A\boldsymbol{p}_n\rangle = 0$, which proves the theorem. □

Theorem 3 *For all* $n \in \mathcal{N}$

$$\beta_n^{PBiCGCR2} = \beta_n^{PBiCR}.$$

Proof. The bi-orthogonality of the search vectors $\boldsymbol{p}_n$ and the pseudo search direction vectors $\hat{\boldsymbol{p}}_n^* = K^{-T}A^T\boldsymbol{p}_n^*$ defined in the inner iteration loop of the PBiCR method yields

$$\begin{aligned} 0 &= \langle K^{-T}A^T\boldsymbol{p}_n^*, A\boldsymbol{p}_{n+1}\rangle = (\boldsymbol{p}_n^*)^T AK^{-1}A\boldsymbol{p}_{n+1} \\ &= (\boldsymbol{p}_n^*)^T AK^{-1}A(K^{-1}\boldsymbol{r}_{n+1} + \beta_n^{PBiCR}\boldsymbol{p}_n) \\ &= (K^{-T}A^T\boldsymbol{p}_n^*)^T AK^{-1}\boldsymbol{r}_{n+1} + \beta_n^{PBiCR}(K^{-T}A^T\boldsymbol{p}_n^*)^T A\boldsymbol{p}_n \\ \Leftrightarrow \qquad \beta_n^{PBiCR} &= -\frac{\langle K^{-T}A^T\boldsymbol{p}_n^*, AK^{-1}\boldsymbol{r}_{n+1}\rangle}{\langle K^{-T}A^T\boldsymbol{p}_n^*, A\boldsymbol{p}_n\rangle} \\ &= \beta_n^{PBiCGCR2} \end{aligned}$$

for all $n = 0, 1, 2, \ldots$, which proves the theorem. □

At this stage, we can establish a general framework for deriving new Lanczos-type iterative solvers: given an initial guess $\boldsymbol{x}_0$ of the solution of the linear system $A\boldsymbol{x} = \boldsymbol{b}$, many methods such as CG, CR, BiCG, BiCOR and BiCR can be unified into the following coupled two–term recurrences by imposing certain conditions [16, 17, 39]:

$$\boldsymbol{r}_0 = \boldsymbol{b} - A\boldsymbol{x}_0, \quad \boldsymbol{p}_0 = \boldsymbol{r}_0, \tag{22}$$

$$\boldsymbol{x}_{j+1} = \boldsymbol{x}_j + \alpha_j\boldsymbol{p}_j, \tag{23}$$

$$\boldsymbol{r}_{j+1} = \boldsymbol{r}_j - \alpha_j A\boldsymbol{p}_j, \tag{24}$$

$$\boldsymbol{p}_{j+1} = \boldsymbol{r}_{j+1} + \beta_j\boldsymbol{p}_j, \qquad \text{for } j = 0, 1, \ldots \tag{25}$$

where $\boldsymbol{r}_j = \boldsymbol{b} - A\boldsymbol{x}_j$ is the j-th residual vector and $\boldsymbol{p}_j$ is the j-th search direction vector. Various formulae used for the parameters α_j, β_j $(j = 0, 1, \ldots)$ in the recurrences (24–25) lead to different algorithms. Denoting by $\mathcal{L}_n$ the underlying constraints subspace, these parameters can be determined by imposing the following orthogonality conditions:

$$\boldsymbol{r}_{j+1} \perp \mathcal{L}_n \text{ and } A\boldsymbol{p}_{j+1} \perp \mathcal{L}_n. \tag{26}$$

For example, $\mathcal{L}_n = \mathcal{K}_n(A, \boldsymbol{r}_0)$ and $\mathcal{L}_n = A\mathcal{K}_n(A, \boldsymbol{r}_0)$ lead respectively to the CG method [11] and the CR method [2, 9] when A is Hermitian positive definite. For non-Hermitian A, the choice $\mathcal{L}_n = \mathcal{K}_n(A^H, \boldsymbol{r}_0^*)$ and$\mathcal{L}_n = \mathcal{K}_n(A^H, A^H\boldsymbol{r}_0^*)$ lead to the BiCG method [12] and the BiCGCR2 method, respectively, while $\mathcal{L}_n = A^H\mathcal{K}_n(A^H, \boldsymbol{r}_0^*)$ leads to the BiCR method [16, 39] and the BiCOR method [17, 18]. Moreover, we have the following condition by the definition of $\mathcal{K}_n$

$$\begin{aligned} \mathcal{L}_n^{\text{BiCGCR2}} &= \mathcal{K}_n(A^H, A^H\boldsymbol{r}_0^*) \\ &= \text{span}\{A^H\boldsymbol{r}_0^*, A^H(A^H\boldsymbol{r}_0^*), \ldots, (A^H)^{n-1}(A^H\boldsymbol{r}_0^*)\} \\ &= A^H \cdot \text{span}\{\boldsymbol{r}_0^*, A^H\boldsymbol{r}_0^*, \ldots, (A^H)^{n-1}\boldsymbol{r}_0^*\} \\ &= A^H\mathcal{K}_n(A^H, \boldsymbol{r}_0^*) = \mathcal{L}_n^{\text{BiCR}}. \end{aligned} \tag{27}$$

To sum up, the BiCGCR2 and BiCR methods indeed possess the same constraints subspace $\mathcal{L}_n$ and mathematical properties (5)-(6), and even their iterative procedures are mostly similar. Taking the preconditioner $M = I$, it is proved that the coefficients α_k and β_k of the BiCGCR2 method and of the BiCR method are *mathematically equivalent*. In general, the BiCGCR2

method often provides the slightly smoother convergence behavior than the BiCR method. The BiCR method, however, appears to be more efficient as it requires one less dot product evaluation at each iteration step and thus saves CPU time. The numerical examples reported in the next section compare the convergence behaviors of both PBiCGCR2 and PBiCR. In addition, Jing et al. [17] had indicated that the BiCOR method is mathematically equivalent to the BiCR method except for a different initial shadow residual. This statement just implies that the BiCOR method is also mathematically equivalent to the BiCGCR2 method except for a different initial shadow residual.

4 Examples and numerical experiments

In this section we demonstrate the potential of the proposed BiCGCR2 method to solve efficiently sparse linear systems, both real and complex. The performance of BiCGCR2 are assessed against the BiCG, BiCR and BiCOR methods, and also against other methods that involve the calculations of the conjugate transpose A^H, such as the popular QMR method. The experiments have been carried out in double precision floating point arithmetic with machine precision 10^{-16} in MATLAB R2014a with a Windows 7 (64 bit) PC-Intel(R) Core(TM) i5-3740 CPU 3.20 GHz, 8 GB of RAM. We measure performance in four aspects: number of iterations (this parameter is referred to as *Iters*), CPU elapsed time in seconds (referred to as *CPU*), $\log_{10}$ of the updated and final true relative residual 2-norms defined respectively as $\log_{10} \|\boldsymbol{r}_n\|_2/\|\boldsymbol{r}_0\|_2$ and $\log_{10} \|\boldsymbol{b} - A\boldsymbol{x}_n\|_2/\|\boldsymbol{r}_0\|_2$ (referred to as *Relres* and *TRR*). Numerical experiments are illustrated by tables of results, but we also plot convergence histories of our runs. The stopping criterion used here is that the 2-norm of the residual must be reduced by a factor (referred to as *TOL*) of the 2-norm of the initial residual, i.e., $\|\boldsymbol{r}_n\|_2/\|\boldsymbol{r}_0\|_2 < tol = 10^{-8}$, or when *Iters* exceeded the maximal iteration number (referred to as *MAXIT*). In all our experiments we take $MAXIT = 6000$.

Example 1 We consider a large set of publicly available linear systems arising from different application areas, and having increasing levels of difficulty, both real nonsymmetric and complex non-Hermitian. We summarize in Table 2 the characteristics of our test matrix problems. The problem denoted as `orsirr_2` is extracted from the Harwell-Boeing collection [46]. The problem denoted as `vdvorst3` arises from solving 2D-problems and is modified from our GitHub source [47]. The problem denoted as `M4D2` arises in computational chemistry is proposed by Sherry Li from NERSC in [48]. The other linear systems are extracted from Tim Davis's matrix collection at the University of Florida [49]. Whenever the physical right-hand side is not available, we use $\boldsymbol{b} = A\boldsymbol{e}$, where $\boldsymbol{e}$ denotes a random vector with entries from -1 to 1. The results of our experiments without preconditioning are reported in Table 3.

We see that the BiCGCR2 method outperforms all of the iterative solvers in terms of number of iterations and CPU time, except the QMR method for `memplus`. However, it is considerably cheaper than the QMR method on this problem and additionally it shows a smaller residual at convergence. On the `epb1` problem, the BiCGCR2 method required about 78% of the iteration steps and computational time of the BiCR method. On the `hcircuit` problem, the BiCGCR2 method converges to the targeted accuracy, whereas the BiCG, BiCOR and QMR methods cannot. It generates more accurate solutions than BiCR on the `pde2961`, `ex36`, `vdvorst3`, `coupled` and `zhao2` problems, and than all other iterative solvers on `vdvorst3`. The experiment

Table 2: Set and characteristics of test matrices in Example 1 (listed in increasing matrix size).

Matrix problem	Reference	Size	Field	$nnz(A)$
orsirr_2	Ref. [46]	886	Oil reservoir simulation	5,970
pde2961	Ref. [49]	2,961	2D/3D problem	14,585
ex36	Ref. [49]	3,079	Computational fluid dynamics	53,099
vdvorst3	Ref. [47]	4,096	2D/3D problem	20,224
rajat13	Ref. [49]	7,598	Circuit simulation problem	48,762
M4D2	Ref. [48]	10,000	Quantum mechanics	127,400
coupled	Ref. [49]	11,341	Circuit simulation problem	97,193
epb1	Ref. [49]	14,734	Thermal problem	95,053
memplus	Ref. [49]	17,758	Circuit simulation problem	99,147
waveguide3D	Ref. [49]	21,036	Electromagnetics problem	303,468
zhao2	Ref. [49]	33,861	Electromagnetics problem	166,453
hcircuit	Ref. [49]	105,676	Circuit simulation problem	513,072

Table 3: The numerical results of different iterative solvers for Example 2.

Method	orsirr_2			pde2961			ex36		
	Iters	*TRR*	CPU	*Iters*	*TRR*	CPU	*Iters*	*TRR*	CPU
BiCGCR2	539	-8.0801	0.0515	230	-8.1259	0.0788	3048	-8.0209	0.6078
BiCR	551	-8.2042	0.0546	248	-8.0477	0.0852	3118	-8.0025	0.6248
BiCG	607	-8.2116	0.0757	246	-8.0956	0.0941	3217	-8.0118	0.8365
BiCOR	626	-8.1493	0.0985	238	-8.2093	0.0792	3416	-8.0370	1.0649
QMR	607	-8.0185	0.0813	251	-8.1104	0.1074	3145	-8.0038	0.9714
Method	vdvorst3			rajat13			M4D2		
	Iters	*TRR*	CPU	*Iters*	*TRR*	CPU	*Iters*	*TRR*	CPU
BiCGCR2	4033	-8.0647	0.7613	358	-8.0231	0.1512	2483	-8.0252	2.7336
BiCR	4289	-8.0400	0.8425	378	-8.1465	0.1569	2512	-8.0506	2.7498
BiCG	5227	-8.0026	1.2965	401	-8.0376	0.1942	2639	-8.1094	3.2753
BiCOR	4207	-8.0262	1.2543	431	-8.0631	0.2306	2525	-8.0596	5.1843
QMR	4805	-8.0010	1.5538	382	-8.2845	0.2478	2583	-8.0167	4.9897
Method	coupled			epb1			memplus		
	Iters	*TRR*	CPU	*Iters*	*TRR*	CPU	*Iters*	*TRR*	CPU
BiCGCR2	923	-8.0807	0.5091	722	-8.2079	0.3713	764	-8.0141	0.4635
BiCR	938	-8.0637	0.5396	923	-8.3901	0.4622	766	-8.0465	0.4806
BiCG	1032	-8.0263	0.6924	-	-7.7974	3.6515	784	-8.0622	0.5879
BiCOR	1102	-8.0886	1.0084	921	-8.3643	0.8089	817	-8.0246	0.8737
QMR	994	-8.0029	0.8957	1066	-8.0120	1.0103	751	-8.0095	0.8478
Method	waveguide3D			zhao2			hcircuit		
	Iters	*TRR*	CPU	*Iters*	*TRR*	CPU	*Iters*	*TRR*	CPU
BiCGCR2	3391	-8.0043	12.5702	1460	-8.0036	1.7671	5875	-8.0015	32.2893
BiCR	3499	-8.0096	13.1258	1495	-8.0004	1.7734	5932	-8.0135	32.4263
BiCG	3651	-8.0280	13.3894	1579	-8.1427	2.2825	-	-7.4282	36.5150
BiCOR	3543	-8.0250	21.1472	1485	-8.0533	3.1877	-	-7.9912	52.1741
QMR	3459	-8.0015	18.2411	1582	-8.0127	3.6996	-	-7.2986	50.0961

also indicate that, as expected, application specific preconditioners may be required to achieve convergence in practice.

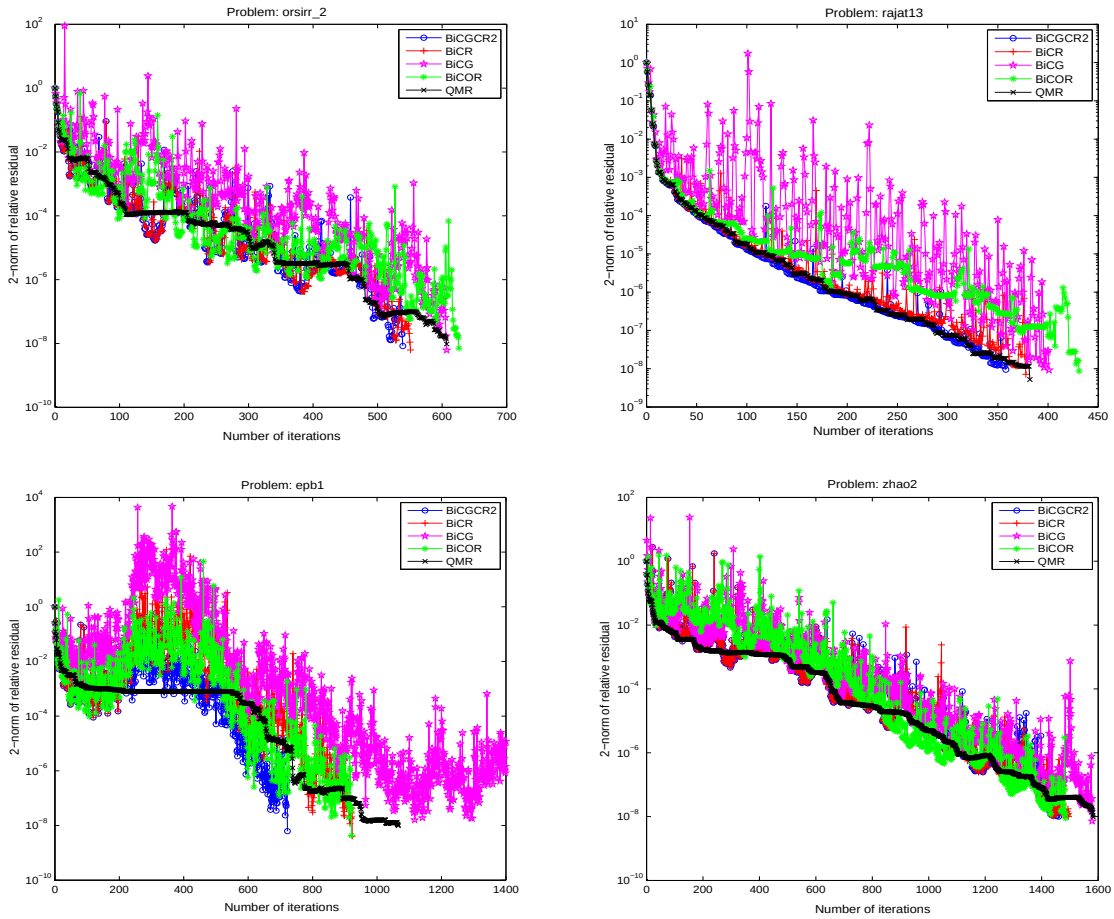

Fig. 1: Convergence histories of different iterative methods for solving different test problems in Example 1.

In Fig. 1, we plot convergence histories of different iterative solvers for the test problems (`orsirr_2`, `rajat13`, `epb1` and `zhao2`)[1]. We observe the typical irregular (oscillating) convergence behaviour of the BiCG method, whereas BiCGCR2, BiCR, BiCOR and QMR exhibit much smoother residual decrease. The convergence curve of the QMR method is the smoothest one, due to the quasi-minimal residual property. The BiCGCR2 method shows smoother convergence curves than both BiCR and BiCOR methods for the test problems (`rajat13`, `epb1` and `zhao2`). In conclusion, our method can be regarded as another efficient iterative solver for dealing with non-Hermitian linear systems.

Example 2 We consider the electromagnetic scattering problem from a large rectangular cavity

[1]For the sake of clarity, we only plot the convergence curve of BiCG method when the iteration step reaches 1400. Because the BiCG method did not meet the required *tol* before $MAXIT$, so the convergence behavior in the last phase is not interesting for us.

represented on the (x, y)-plane. We assume that the medium is y-directional inhomogeneous, and we consider the transverse magnetic polarization case. The model Helmholtz equation with positive wave number is discretized by the five-point finite difference scheme with uniform stepsize h, leading to a nonsymmetric system of linear equations of the following form

$$A\boldsymbol{u} = \boldsymbol{b}, \quad A = \begin{pmatrix} B & E \\ F & C \end{pmatrix},$$

where the sub-matrices are defined as follows,

$$B = V \otimes I + I \otimes V \in \mathbb{R}^{p\times p}, \quad C = I - hG \in \mathbb{R}^{q\times q}, \quad E = I \otimes \boldsymbol{e}_q \in \mathbb{R}^{p\times q}$$

and $F = -E^T$, where $h = \frac{1}{q+1}, p = q^2, \theta \geq 0$ is a real constant, $\boldsymbol{e}_q$ is the q-th unit vector in $\mathbb{R}^q$, I is the q-by-q identity matrix, $V = \text{tridiag}(-1 + \frac{\theta h}{2}, 2, -1 - \frac{\theta h}{2}) \in \mathbb{R}^{q\times q}$ is a tridiagonal matrix, $\Omega = h^2 \cdot \text{diag}(\omega_1^2, \omega_2^2, \ldots, \omega_q^2) \in \mathbb{R}^{q\times q}$ is a nonnegative diagonal matrix, $G = (g_{ij}) \in \mathbb{R}^{q\times q}$, and $\otimes$ denotes the Kronecker product; we refer the reader to [50] for details. In our computations we take $\theta = 1$ and $g_{ij} = \frac{1}{(i+j)^2}$. For simplicity, the linear system is defined via choosing a discrete solution $\boldsymbol{u}$ consisting of uniformly distributed random numbers in the interval $[-1, 1]$, and the right-hand side is then computed as $\boldsymbol{b} = A\boldsymbol{u}$. Numerical results with different iterative solvers are reported in the following Table 4.

Table 4: The numerical results of different iterative solvers for Example 2.

Method	$(q = 40,\ \omega_i = 8\pi)$			$(q = 50,\ \omega_i = 10\pi)$			$(q = 60,\ \omega_i = 12\pi)$		
	Iters	*TRR*	CPU	*Iters*	*TRR*	CPU	*Iters*	*TRR*	CPU
BiCGCR2	749	-8.0600	0.0762	1218	-8.1417	0.1957	2858	-8.0348	0.5303
BiCR	776	-8.2382	0.0837	1276	-8.1101	0.2039	3202	-8.1316	0.5918
BiCG	781	-8.1404	0.1142	1450	-8.0277	0.3008	3846	-8.1754	0.9256
BiCOR	828	-8.1821	0.1518	1308	-8.1896	0.2840	2927	-8.0927	0.8183
QMR	781	-5.0213	0.1323	1358	-8.0085	0.3461	3379	-8.0179	1.0225
Method	$(q = 70,\ \omega_i = 14\pi)$			$(q = 80,\ \omega_i = 16\pi)$			$(q = 90,\ \omega_i = 18\pi)$		
	Iters	*TRR*	CPU	*Iters*	*TRR*	CPU	*Iters*	*TRR*	CPU
BiCGCR2	2418	-8.1644	0.5476	3420	-8.0039	0.8557	3809	-8.0266	1.1268
BiCR	2651	-8.0354	0.5857	3565	-8.0247	0.9105	3974	-8.2344	1.1743
BiCG	2738	-8.0746	0.8096	3916	-8.0013	1.3083	3941	-8.0009	1.5529
BiCOR	2569	-8.0192	0.9024	4603	-8.0398	1.8225	3980	-8.0292	1.9378
QMR	2455	-8.0218	0.9441	3603	-8.0138	1.6294	4065	-8.0044	2.3321

The BiCGCR2 method has the best performance among all of the iterative solvers in terms of number of iterations and CPU time. The QMR method exhibits smoother convergence due to its quasi-minimal residual property. However, this method and the BiCOR method are considerably more expensive than BiCGCR2, BiCR and BiCG methods in terms of CPU time. The BiCGCR2 method generated better approximate solutions than all other iterative solvers in the case $q = 70, \omega_i = 14\pi$. By the way, as shown by our experiments, specific preconditioners may be required for accelerating the convergence on the Helmholtz equation [51].

Convergence histories of different iterative solvers for the case $q = 40, \omega_i = 8\pi$ and $q = 50, \omega_i = 10\pi$ are plotted in Fig. 2. We see that the BiCG method displays its typically oscillating convergence behaviour, whereas BiCGCR2, BiCR, BiCOR and QMR have much smoother convergence. The QMR method is the smoothest one among these five iterative solvers. The

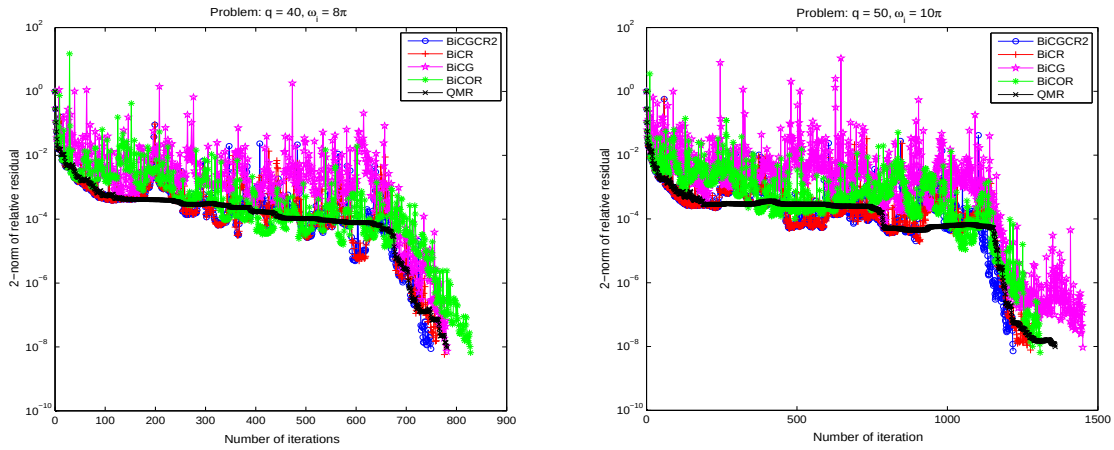

Fig. 2: Convergence histories of different iterative methods for solving different test problems in Example 2.

BiCGCR2 method has smoother convergence than both BiCR and BiCOR methods for the case $q = 50, \omega_i = 10\pi$. We conclude that the BiCGCR2 method can be considered an efficient alternative to other iterative solvers for this test problem.

Example 3 Finally, we test the new proposed Krylov method in combination with preconditioning on a set of publicly available linear systems arising from different application areas; these systems are extracted from Tim Davis's matrix collection available at the University of Florida. We consider both real nonsymmetric and complex non-Hermitian linear systems. We summarize in Table 5 the characteristics of the linear systems that were solved. When a physical right-hand side (referred to as *RHS*) is not available, we use $\boldsymbol{b} = A\boldsymbol{e}$, where $\boldsymbol{e}$ is a random vector with entries from -1 to 1[2] Here we assess the performance of BiCGCR2 and other iterative solvers in combination with the ILU(0) preconditioning [2, pp. 307-310]. For stability reasons, we compute an ILU(0) factorization of $A + \sigma I$, where $\sigma = 10^{-12}$ if all diagonal elements of A are zero, or $\sigma = 10^{-12} \max\{|a_{ii}|\}$ if some but not all diagonal elements a_{ii} of A zero, or $\sigma = 0$ otherwise. This procedure follows recommendations in [52]. The numerical results obtained from different iterative solvers with ILU(0) preconditioning are shown in Table 6.

The results indicate that the PBiCGCR2 method performs better than the other preconditioned iterative solvers in terms of number of iterations and CPU time. Once again, the preconditioned QMR method (denoted as PQMR) exhibits smoother convergence because of its quasi-minimal residual property but is more expensive than BiCGCR2, BiCR and BiCG methods with ILU(0) preconditioners in terms of CPU time. In addition, the PBiCGCR2 method achieves the best final accuracy than all of the other preconditioned iterative solvers on the `epb3` problem. We see from the results on the `rajat12`, `epb1`, `memplus` and `epb3` problems that the preconditioned BiCOR (denoted as PBiCOR) method is sometimes expensive to use. The convergence performance of PBiCR method is greatly similar with the PBiCGCR2 and PBiCOR methods in aspects of the number of iterations, CPU time and *TRR*, which is in agreement with

[2]In order to investigate the different problems, here the *RHS* used in Example 3 is different from that defined in Example 1.

Table 5: Set and characteristics of test matrices in Example 3 (listed in increasing matrix size).

Matrix problem	Reference	Size	Field	$nnz(A)$
watt_2	Ref. [49]	1,856	Computational fluid dynamics	11,550
rajat12	Ref. [49]	1,879	Circuit simulation problem	12,818
ex31	Ref. [49]	3,909	Computational fluid dynamics	91,223
ex40	Ref. [49]	7,740	Computational fluid dynamics	456,188
Grond1e4	Ref. [47]	10,000	Computational fluid dynamics	49,600
epb1	Ref. [49]	14,734	Thermal problem	95,053
memplus	Ref. [49]	17,758	Circuit simulation problem	99,147
Grond4e4	Ref. [47]	40,000	Computational fluid dynamics	199,200
epb3	Ref. [49]	84,617	Thermal problem	463,625

Table 6: Numerical results of different iterative solvers with ILU(0) preconditioning for Example 3.

Method	watt_2			rajat12			ex31		
	Iters	*TRR*	CPU	*Iters*	*TRR*	CPU	*Iters*	*TRR*	CPU
BiCGCR2	55	-8.0954	0.0281	78	-8.0021	0.0384	127	-8.0245	0.1941
BiCR	57	-8.2757	0.0299	81	-8.2387	0.0412	133	-8.0648	0.2186
BiCG	57	-8.0014	0.0413	80	-8.0168	0.0495	134	-8.1005	0.2439
BiCOR	57	-8.2898	0.0441	82	-8.1529	0.0571	131	-8.1504	0.2167
QMR	57	-8.6045	0.0521	81	-8.2381	0.0543	135	-8.0074	0.2468
Method	ex40			Grond1e4			epb1		
	Iters	*TRR*	CPU	*Iters*	*TRR*	CPU	*Iters*	*TRR*	CPU
BiCGCR2	133	-8.0750	1.4552	193	-8.0096	0.2541	128	-9.1420	0.3261
BiCR	136	-8.3157	1.4868	196	-8.3503	0.2872	130	-8.1316	0.3601
BiCG	152	-8.0137	1.6475	195	-8.0013	0.3229	129	-8.0424	0.3901
BiCOR	137	-8.1947	1.6330	196	-8.1693	0.3700	131	-8.6027	0.4728
QMR	140	-8.1176	1.6241	195	-8.0547	0.3924	129	-8.1306	0.4617
Method	memplus			Grond4e4			epb3		
	Iters	*TRR*	CPU	*Iters*	*TRR*	CPU	*Iters*	*TRR*	CPU
BiCGCR2	185	-8.0408	0.5678	411	-8.2541	2.9472	146	-8.6211	3.0328
BiCR	186	-8.0362	0.5765	441	-8.3665	3.1191	187	-8.1432	3.8747
BiCG	202	-8.0906	0.7065	441	-8.3168	3.2796	178	-8.0306	3.8680
BiCOR	193	-8.0437	0.8210	413	-8.0931	3.7033	174	-8.2180	4.2685
QMR	186	-8.0295	0.7832	441	-8.2795	3.8781	180	-8.1139	4.2240

the theoretical results.

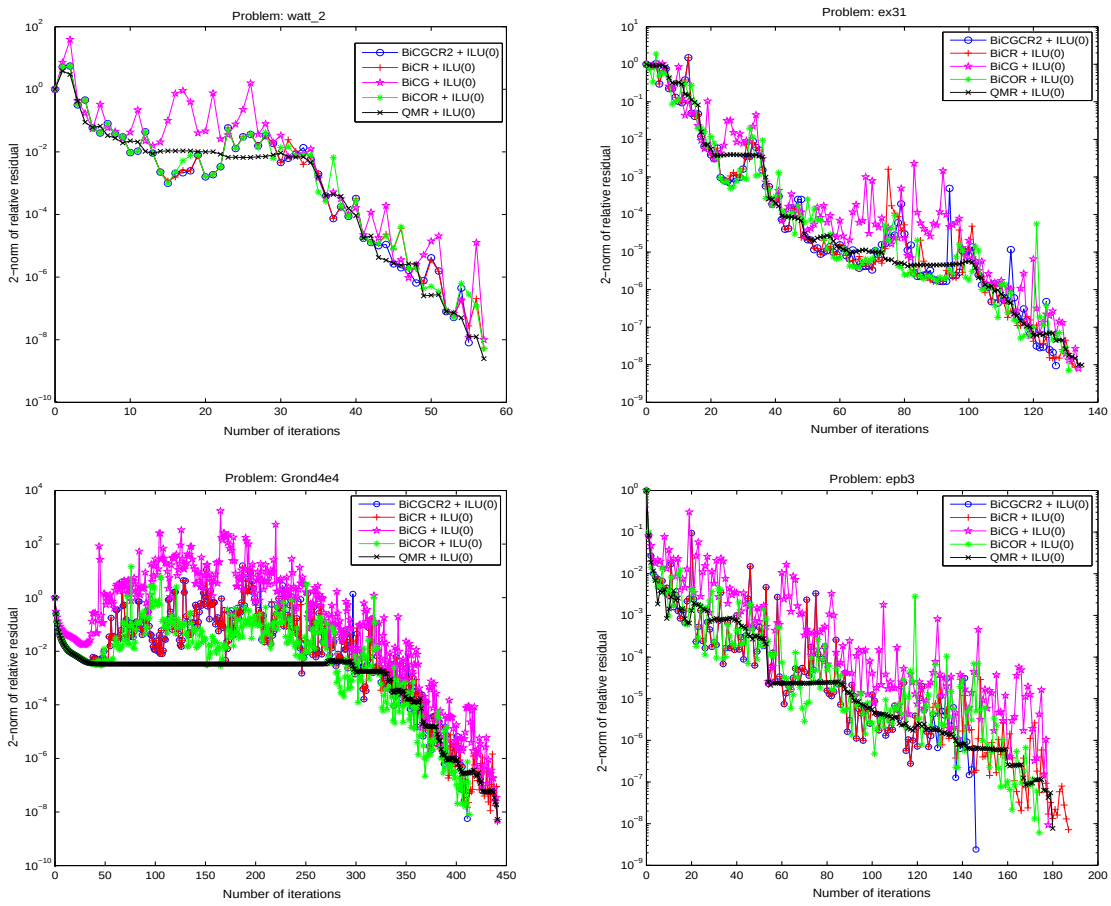

Fig. 3: Convergence histories of different iterative methods with ILU(0) preconditioning for solving different test problems in Example 3.

In Fig. 3 we plot convergence histories of different iterative solvers with the ILU(0) preconditioner for the test problems denoted as (`watt_2`, `ex31`, `Grond4e4` and `epb3`). We see that the convergence behavior with the preconditioned BiCG (PBiCG) was still jagged, whereas those with the PBiCGCR2, PBiCR, PBiCOR and PQMR methods were smoother. Moreover, due to the quasi-minimal residual property, the convergence curve of PQMR method is the smoothest one among these five preconditioned iterative solvers. The PBiCGCR2, PBiCR and PBiCOR methods displayed indeed similar convergence behaviors in the start phase, whereas the PBiCGCR2 method shown considerably attractive convergence behavior in the latter convergence phase, especially for `epb3` problem. The PBiCGCR2 method even provided smoother convergence curves than both PBiCR and PBiCOR methods for the test problems (i.e., `watt_2` and `epb3`). In summary, the PBiCGCR2 method can be considered as efficient as the other preconditioned iterative solvers for handling the targeted linear systems.

5 Conclusions

Starting from the pioneering work on two families of iterative solvers, i.e., the COCR/BiCGCR and the BiCR/BiCOR methods, in this paper we propose a new extension of CR for solving non-Hermitian linear systems, which is still based on short-term vector recurrences. The resulting algorithm, named BiCGCR2, reduces to CR if the coefficient matrix A is Hermitian. We have described the complete derivation of the BiCGCR2 algorithm (also including PBiCGCR2) for non-Hermitian linear systems and have investigated the relation among CR, BiCGCR, COCR, BiCR and BiCOR. Moreover, we also proved that the proposed method (BiCGCR2) is mathematically equivalent to BiCR and BiCOR. Extensive numerical examples are reported to assess the performance of our proposed method also against other established iterative solvers. The theoretical findings and the numerical results indicates that the proposed method can be viewed as an efficient tool for solving non-Hermitian linear systems arising in numerical applications.

The numerical experiments have revealed that BiCGCR2 tends to show smoother convergence behavior and often faster convergence than BiCG, BiCR and BiCOR for some practical applications. Therefore it can be used as a basic iterative procedure for the development of other non-optimal Krylov subspace methods, similarly to the BiCG, BiCR and BiCOR algorithm that have motivated the development of BiCGSTAB(ℓ) (GPBiCG) [23, 25], BiCRSTAB(ℓ) (GPBiCR) [16, 32] and BiCORSTAB2 (GPBiCOR) [34, 35, 53]. In future work, we plan to construct hybrid variants of BiCGCR2, for which $\boldsymbol{r}_n := H_n(A)\boldsymbol{r}_n^{BiCGCR2}$ where H_n is a suitable matrix polynomial of degree n, along the same lines of the derivations of hybrid BiCG, hybrid BiCR or hybrid BiCOR.

Acknowledgements

The authors would like to thank Prof. Markus Clemens for his helpful and insightful discussions. We are also grateful to the anonymous referees and editor Prof. Michael Ng for their useful suggestions and comments that improved the presentation of this paper. This research is supported by 973 Program (2013CB329404), NSFC (61370147, 61170311, 61402082, and 11301057), the Fundamental Research Funds for the Central Universities (ZYGX2013J106, ZYGX2013Z005, and ZYGX2014J084).

References

[1] T.A. Davis, Direct Methods for Sparse Linear Systems, SIAM Series on the Fundamentals of Algorithms, SIAM, Philadelphia, USA, 2006.

[2] Y. Saad, Iterative Methods for Sparse Linear Systems, second ed., SIAM, Philadelphia, USA, 2003.

[3] V. Simoncini, D.B. Szyld, Recent computational developments in Krylov subspace methods for linear systems, Numer. Linear Algebra Appl., 14 (2007), pp. 1-59.

[4] M.H. Gutknecht, Lanczos-type solvers for nonsymmetric linear systems of equations, Acta Numer., 6 (1997), pp. 271-397.

[5] R. Barrett, M. Berry, T.F. Chan, J. Demmel, J. Donato, J. Dongarra, V. Eijkhout, R. Pozo, C. Romine, H.A. van der Vorst, Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods (2nd Edition), SIAM, Philadelphia, USA, 1994.

[6] M.R. Hestenes, E. Stiefel, Methods of conjugate gradients for solving linear systems, J. Res. Nat. Bur. Standards, 49 (1952), pp. 409-436.

[7] X.-M. Gu, M. Clemens, T.-Z. Huang, L. Li, The SCBiCG class of algorithms for complex symmetric systems with applicationsin several electromagnetic model problems, Comput. Phys. Commun., 191 (2015), pp. 52-64.

[8] C. Paige, M. Saunders, Solution of sparse indefinite systems of linear equations, SIAM J. Numer. Anal. 12 (1975), pp. 617-629.

[9] E. Stiefel, Relaxationsmethoden bester Strategie zur Lösung linearer Gleichungssysteme, Comment. Math. Helv., 29 (1955), pp. 157-179.

[10] S.F. Ashby, T.A. Manteuffel, P.E. Saylor, A taxonomy for conjugate gradient methods, SIAM J. Numer. Anal., 27 (1990), pp. 1542-1568.

[11] C. Lanczos, Solution of systems of linear equations by minized itertions, J. Res. Nat. Bureau Standards, 49 (1952), pp. 33-53.

[12] R. Fletcher, Conjugate gradient methods for indefinite systems, in *Numerical Analysis-Dundee 1975*, Alistair Watson G. (ed.), Lecture Notes in Mathematics, vol. 506, Springer: Heidelberg, 1976, pp. 73-89.

[13] Y. Saad, M.H. Schultz, GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems, SIAM J. Sci. Stat. Comput., 7 (1986), pp. 856-869.

[14] S.C. Eisenstat, H.C. Elman, M.H. Schultz, Variational iterative methods for nonsymmeric systems of linear equations, SIAM J. Sci. Numer. Anal., 20 (1983), pp. 345537.

[15] T. Sogabe, M. Sugihara, S.-L. Zhang, An extension of the conjugate residual method to nonsymmetric linear systems, J. Comput. Appl. Math., 226 (2009), pp. 103-113.

[16] T. Sogabe, Extensions of the conjugate residual method (Ph.D. dissertation), Department of Applied Physics, University of Tokyo, Tokyo, Japan, 2006. Avaiable online at `http://www.ist.aichi-pu.ac.jp/person/sogabe/thesis.pdf`.

[17] Y.-F. Jing, T.-Z. Huang, Y. Zhang, L. Li, G.-H. Cheng, Z.-G. Ren, Y. Duan, T. Sogabe, B. Carpentieri, Lanczos-type variants of the COCR method for complex nonsymmetric linear systems, J. Comput. Phys., 228 (2009), pp. 6376-6394.

[18] B. Carpentieri, Y.-F. Jing, T.-Z. Huang, The BiCOR and CORS iterative algorithms for solving nonsymmetric linear systems, SIAM J. Sci. Comput., 33 (2011), pp. 3020-3036.

[19] P. Sonneveld, CGS, a fast Lanczos-type solver for nonsymmetric linear systems, SIAM J. Sci. Stat. Comput., 10 (1989), pp. 36-52.

[20] H.A. van der Vorst, Bi-CGSTAB: A fast and smoothly converging variant of Bi-CG for the solution of nonsymmetric linear systems, SIAM J. Sci. Stat. Comput., 13 (1992), pp. 631-644.

[21] M.H. Gutknecht, Variants of BiCGSTAB for matrices with complex spectrum, SIAM J. Sci. Comput, 14 (1993), pp. 1020-1033.

[22] G.L.G. Sleijpen, H.A. van der Vorst, D.R. Fokkema, BiCGstab(ℓ) and other hybrid Bi-CG methods, Numer. Algorithms, 7 (1994), pp. 75-109.

[23] G.L.G. Sleijpen, D. R. Fokkema, BiCGstab(ℓ) for linear equations involving unsymmetric matrices with complex spectrum, Electron. Trans. Numer. Anal., 1 (1993), pp. 11-32.

[24] D.R. Fokkema, G.L.G. Sleijpen, H.A. Van der Vorst, Generalized conjugate gradient squared, J. Comput. Appl. Math., 71 (1996), pp. 125-146.

[25] S.-L. Zhang, GPBi-CG: Generalized product-type methods based on Bi-CG for solving nonsymmetric linear systems, SIAM J. Sci. Comput., 18 (1997), pp. 537-551.

[26] R.W. Freund, N.M. Nachtigal, QMR: A quasi-minimal residual method for non-Hermitian linear systems, Numer. Math., 60 (1991), pp. 315-339.

[27] R.W. Freund, M.H. Gutknecht, N.M. Nachtigal, An implementation of the look-ahead Lanczos algorithm for non-Hermitian matrices, SIAM J. Sci. Comput., 14 (1993), pp. 137-158.

[28] R.W. Freund, A transpose-free quasi-minimal residual algorithm for non-Hermitian linear systems, SIAM J. Sci. Comput., 14 (1993), pp. 470-482.

[29] T.F. Chan, E. Gallopoulos, V. Simoncini, T. Szeto, C.H. Tong, A quasi-minimal residual variant of the Bi-CGSTAB algorithm for nonsymmetric systems, SIAM J. Sci. Comput., 15 (1994), pp. 338-347.

[30] P. Sonneveld, M.B. van Gijzen, IDR(s): A family of simple and fast algorithms for solving large nonsymmetric linear systems. SIAM J. Sci. Comput., 31 (2008), pp. 1035-1062.

[31] M.-C. Yeung, T.F. Chan, ML(k)BiCGSTAB: A BiCGSTAB variant based on multiple Lanczos starting vectors, SIAM J. Sci. Comput., 21 (1999), pp. 1263-1290.

[32] K. Abe, G.L.G. Sleijpen, BiCR variants of the hybrid BiCG methods for solving linear systems with nonsymmetric matrices, J. Comput. Appl. Math., 234 (2010), pp. 985-994.

[33] J. Zhang, H. Dai, Generalized conjugate A-orthogonal residual squared method for complex non-Hermitian linear systems, J. Comput. Math., 32 (2014), pp. 248-265.

[34] L. Zhao, T.-Z. Huang, A hybrid variant of the BiCOR method for a nonsymmetric linear system with a complex spectrum, Appl. Math. Lett. 26 (2013), pp. 457-462.

[35] L. Zhao, T.-Z. Huan, Y.-F. Jing, L.-J. Deng, A generalized product-type BiCOR method and its application in signal deconvolution, Comput. Math. Appl., 66 (2013), pp. 1372-1388.

[36] Y.-F. Jing, T.-Z. Huang, Y. Duan, B. Carpentieri, A comparative study of iterative solutions to linear systems arising in quantum mechanics, J. Comput. Phys., 229 (2010), pp. 8511-8520.

[37] R.W. Freund, T. Szeto, A quasi-minimal residual squared algorithm for non-Hermitian linear systems, in *Proceeding of 2nd Copper Mountain Iterative Methods Conference*, April 1992, UCLA-CAM Tech. Rep. 92-19. Avaiable online at `ftp://ftp.math.ucla.edu/pub/camreport/cam92-19.pdf`.

[38] J. Zhang, H. Dai, A new quasi-minimal residual method based on a biconjugate A-orthonormalization procedure and coupled two-term recurrences, Numer. Algorithms, 70 (2015), pp. 875-896.

[39] T. Sogabe, S.-L. Zhang, A COCR method for solving complex symmetric linear systems J. Comput. Appl. Math., 199 (2007), pp. 297-303.

[40] M. Clemens, T. Weiland, U. van Rienen, Comparison of Krylov-type methods for complex linear systems applied to high-voltage problems, IEEE Trans. Magn., 34 (5) (1998), pp. 3335-3338.

[41] M. Clemens, U. van Rienen, T. Weiland, Correction to "comparison of Krylov-type methods for complex linear systems applied to high-voltage problems", IEEE Trans. Magn., (5) (50), 2014, p. 9700101.

[42] J. Zhang, H. Dai, J. Zhao, A new family of global methods for linear systems with multiple right-hand sides, J. Comput. Appl. Math., 236 (2011), pp. 1562-1575.

[43] X.-M. Gu, T.-Z. Huang, J. Meng, T. Sogabe, H.-B. Li, L. Li, BiCR-type methods for families of shifted linear systems, Comput. Math. Appl., 68 (2014), pp. 746-758.

[44] T.-X. Gu, X.-Y. Zuo, L.-T. Zhang, W.-Q. Zhang, Z.-Q. Sheng, An improved bi-conjugate residual algorithm suitable for distributed parallel computing, Appl. Math. Comput., 186 (2007), pp. 1243-1253.

[45] M. Clemens, T. Weiland, Iterative methods for the solution of very large complex-symmetric linear systems of equations in electromagnetics, in *Eleventh Copper Mountain Conference on Iterative Methods*, Part 2, T.A. Manteuffel, S.F. McCormick (Eds.), 1996, 7 pages.

[46] I.S. Duff, R.G. Grimes, J.G. Lewis, User's guide for the Harwell-Boeing sparse matrix collection, Tech. Rep. RAL-92-086, Rutherford Appleton Lab., Chilton, UK, 1992.

[47] X.-M. Gu, GitHub's repositories: Test_matrices, Augest 2015. Available online at `https://github.com/Hsien-Ming-Ku/Test_matrices/tree/master/Problems`.

[48] M. Baertschy, X. Li, Solution of a three-body problem in quantum mechanics using sparse linear algebra on parallel computers, in *Proceedings of the 2001 ACM/IEEE conference on Supercomputing (Supercomputing '01)*, ACM, New York, 2001, p. 47.

[49] T. Davis, Y. Hu, The University of Florida Sparse Matrix Collection, ACM Trans. Math. Softw., 38 (1) (2011), Article 1, 25 pages. Avaiable online at `http://www.cise.ufl.edu/research/sparse/matrices/`.

[50] Z.-Z. Bai, Structured preconditioners for nonsingular matrices of block two-by-two structures, Math. Comp., 75 (2006), pp. 791-815.

[51] O.G. Ernst, M.J. Gander, Why it is difficult to solve Helmholtz problems with classical iterative methods, in *vol. 83 of Numerical Analysis of Multiscale Problems*. Edited by I. Graham, T. Hou, O. Lakkis and R. Scheichl. Springer-Verlag (2011), pp. 325-361.

[52] E. Chow, Y. Saad, Experimental study of ILU preconditioners for indefinite matrices, J. Comput. Appl. Math., 86 (1997), pp. 387-414.

[53] X.-M. Gu, T.-Z. Huang, B. Carpentieri, L. Li, C. Wen, A hybridized iterative algorithm of the BiCORSTAB and GPBiCOR methods for solving non-Hermitian linear systems, Comput. Math. Appl., 70 (2015), pp. 3019-3031.